



# Apresentação do Relacionamento entre Entidades e sua aplicação no Spring Boot

Antes de vermos alguns exemplos de como as relações entre entidades são implementadas no Spring Boot, é importante reforçarmos alguns conceitos relacionados. Um bom ponto de partida é observarmos alguns exemplos e definições utilizadas em disciplinas direcionadas ao Banco de Dados. Vamos começar com a imagem abaixo:

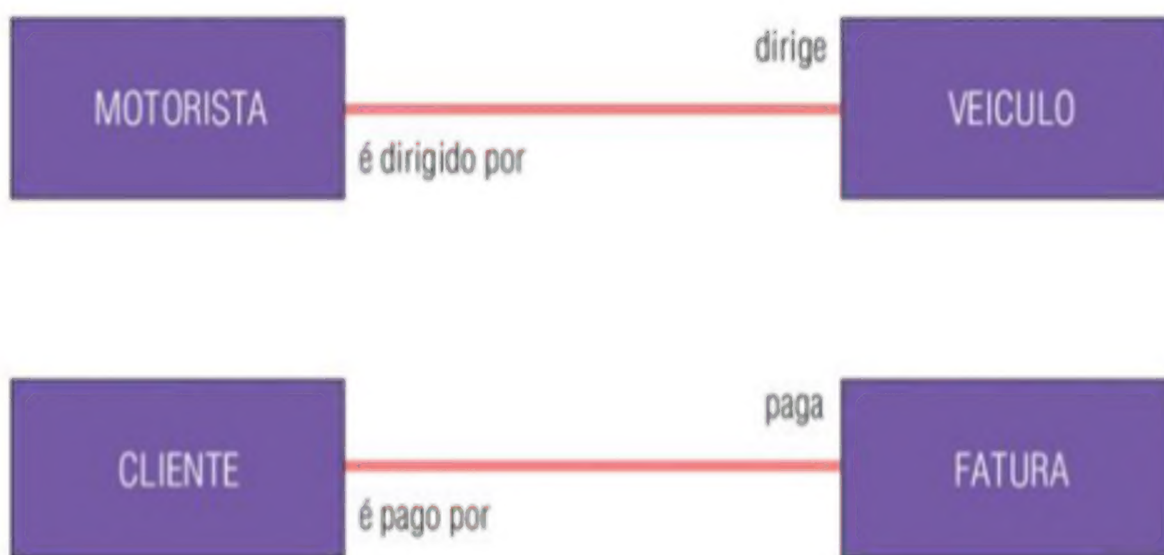



Figura 1: Relacionamento entre Entidades



É possível perceber algumas relações estabelecidas entre diferentes entidades. Poderíamos, usando uma linguagem

associada ao banco de dados, dizer que:



- Um motorista dirige um veículo. Um veículo é dirigido por um motorista.

- Um cliente paga uma fatura. Uma fatura é paga por um cliente.

OBS: As palavras sublinhadas representam as entidades nos relacionamentos em questão.

Repare que, nas relações descritas acima, o quantitativo utilizado para expressar a relação é o número 1(um), ou seja, ainda considerando o contexto apresentado, poderíamos dizer que um veículo será sempre dirigido por apenas um motorista. Da mesma forma, poderíamos dizer que um motorista só poderá dirigir um veículo. Essa quantificação, aplicada aos relacionamentos entre entidades, é chamada, ainda na disciplina de banco de dados, de **cardinalidade**. Guarde esse nome, pois ele poderá ser utilizado também nas disciplinas de desenvolvimento, sempre com essa finalidade: expressar, em termos quantitativos, como os dados de uma entidade são associados aos dados de outra entidade. Um outro ponto importante para se ter em mente é que a cardinalidade é restrita ao escopo de negócio. Não existe cardinalidade que se aplique a todos os tipos ou negócios em si. Na prática, isso significa dizer que uma cardinalidade entre duas entidades iguais podem ser diferentes em diferentes negócios/empresas.

Tendo visto os conceitos iniciais sobre cardinalidade, podemos abordar alguns tipos de relacionamento e sua aplicação prática. Veremos, inicialmente, a relação “one-to-one” e, a seguir, a “one-to-many”.

Na relação “one-to-one” ou um(a)-para-um(a), uma ocorrência da entidade X só poderá se relacionar com somente uma ocorrência da entidade Y. Os exemplos vistos na figura 1 representam esse tipo de relacionamento. Podemos pensar ainda em outras situações em que tal relacionamento se faz presente, dentro de determinado negócio/sistema/software:

- Um cliente só pode ter um número de telefone e um número de telefone só pode estar associado a um cliente. Logo, um mesmo número de telefone não pode estar vinculado a mais de um cliente, assim como um cliente não pode ter mais de um número de telefone.

- Um professor só pode lecionar uma disciplina. Uma disciplina só pode ser lecionada por um professor. Logo, um professor não pode lecionar mais de uma disciplina e uma disciplina só pode ser lecionada por um professor.

Utilizando uma notação típica de DER (Diagrama Entidade Relacionamentos), podemos representar o primeiro exemplo acima da seguinte forma:




Figura 2: Relacionamento one-to-one



No Spring Boot, o relacionamento demonstrado acima implicaria na criação de duas Entities (Entidades): Cliente e Telefone e seus respectivos atributos. Tal processo, conhecido como Mapeamento Objeto-Relacional (ORM), consiste na criação de classes Java contendo atributos, métodos e anotações – que são a forma como o Spring Boot identifica tanto o papel da classe, ou seja, representar uma Entidade, como os atributos e as colunas do banco de dados às quais se referem e, por fim, o relacionamento. A seguir, os códigos das entidades, no padrão Spring Boot, Cliente e Telefone são demonstrados.

De antemão, seguem algumas explicações extras para ajudar na melhor compreensão do código:

- O Spring Boot utiliza o padrão de anotação para identificar elementos do código. Uma anotação é composta por uma palavra reservada precedida pelo símbolo @ . São exemplos de anotação nos códigos a seguir: @Entity (identifica que a classe é uma Entidade); @Id (identifica que um atributo tem o papel de identificador da classe – papel semelhante ao da chave-primária na tabela do banco de dados);

- A relação one-to-one é implementada através da anotação “OneToOne”. Repare que as duas Entidades recebem tal anotação – porém, com pequenas diferenças;

- O Spring tem anotações próprias e também faz uso de anotações de outras bibliotecas, como a JPA (as anotações de uma entidade são, em sua maioria, provenientes da JPA).



```
@Entity
@Table(name="cliente")
public class Cliente{
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "cliente_id")
   private Integer clienteId;

   @Column(name = "cliente_nome")
   private String clienteNome;

   @OneToOne
   @JoinColumn(name = "telefone_id", referencedColumnName = "telefone_id")
   private Telefone telefone;

   public Integer getClienteId() {
      return clienteId;
   }

   public void setClienteId(Integer clienteId) {
      this.clienteId = clienteId;
   }

   public String getClienteNome() {
      return clienteNome;
   }

   public void setClienteNome(String clienteNome) {
      this.clienteNome = clienteNome;
   }

   public Telefone getTelefone() {
      return telefone;
   }

   public void setTelefone(Telefone telefone) {
      this.telefone = telefone;
   }
}
```

```
@Entity
@Table(name="cliente")
public class Cliente{
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "cliente_id")
   private Integer clienteId;

   @Column(name = "cliente_nome")
   private String clienteNome;

   @OneToOne
   @JoinColumn(name = "telefone_id", referencedColumnName = "telefone_id")
   private Telefone telefone;

   public Integer getClienteId() {
      return clienteId;
   }

   public void setClienteId(Integer clienteId) {
      this.clienteId = clienteId;
   }

   public String getClienteNome() {
      return clienteNome;
   }

   public void setClienteNome(String clienteNome) {
      this.clienteNome = clienteNome;
   }

   public Telefone getTelefone() {
      return telefone;
   }

   public void setTelefone(Telefone telefone) {
      this.telefone = telefone;
   }
}
```

Agora, vamos falar sobre o relacionamento one-to-many. Como o nome indica, neste relacionamento temos uma instância da entidade A podendo se relacionar com várias entidades da entidade B. Além disso, tais instâncias na entidade B só podem se relacionar com a instância a ela associada na entidade A. Veja um exemplo desse tipo de relação no DER abaixo:

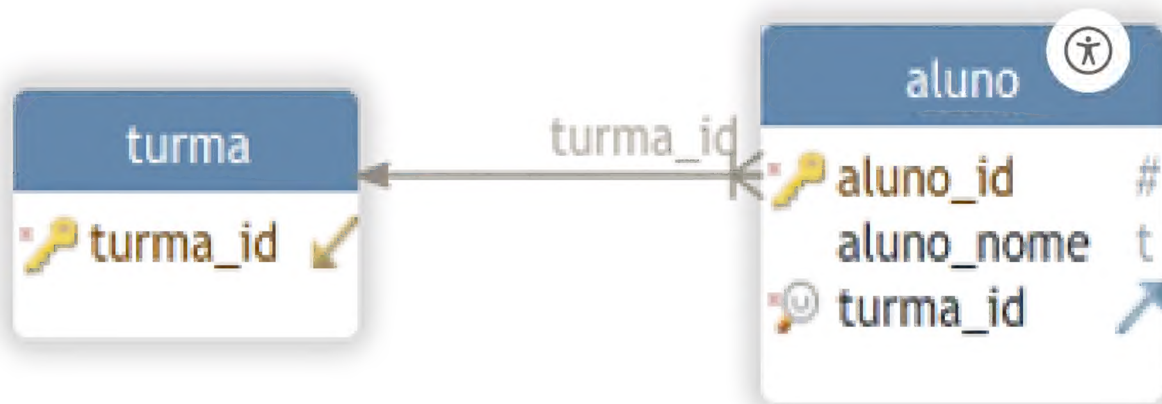



Figura 3: Relacionamento one-to-many

Analisando o diagrama acima, temos a seguinte relação e cardinalidade: uma turma é composta por vários alunos e um aluno só pode estar vinculado a uma turma. No Spring Boot essa relação é mapeada da seguinte forma:



```
@Entity
@Table(name="turma")
public class Turma{
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "turma_id")
   private Integer turmaId;

   @OneToMany(mappedBy="turma")
   private Set<Turma> turmas;

   public Integer getTurmaId() {
      return turmaId;
   }

   public void setTurmaId(Integer turmaId) {
      this.turmaId = turmaId;
   }

   public Set<Turma> getTurmas() {
      return turmas;
   }

   public void setTurmas(Set<Turma> turmas) {
      this.turmas = turmas;
   }
}
```



```
@Entity
@Table(name="aluno")
public class Aluno{
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "aluno_id")
   private Integer alunoId;

   @Column(name = "aluno_nome")
   private String alunoNome;

   @ManyToOne
   @JoinColumn(name="turma_id")
   private Turma turma;

   public Integer getAlunoId() {
      return alunoId;
   }

   public void setAlunoId(Integer alunoId) {
      this.alunoId = alunoId;
   }

   public String getAlunoNome() {
      return alunoNome;
   }

   public void setAlunoNome(String alunoNome) {
      this.alunoNome = alunoNome;
   }

   public Turma getTurma() {
      return turma;
   }

   public void setTurma(Turma turma) {
      this.turma = turma;
   }
}
```

Comparando os códigos dessas duas últimas classes/entidades, você vai reparar pequenas diferenças em relação às duas primeiras: aqui temos uma anotação específica para o relacionamento one-to-many: OneToMany e outra pra relação inversa, o ManyToOne. Inclusive, sobre essa particularidade, cabe destacar que no Spring é possível fazer o relacionamento bidirecional, ou seja, é possível listar todos os alunos pertencentes a uma turma, através da entidade Turma, e também a turma à qual o aluno está vinculado, através da entidade Aluno. Por último, uma

outra diferença importante no relacionamento OneToMany é que, em Turma, não temos apenas uma instância de Aluno, mas sim uma coleção – que no exemplo é representado pelo “Set<Turma>”.

Portanto, revisamos alguns conceitos de bancos de dados, como os relacionamentos e cardinalidade, necessários para a criação das Entidades no framework Spring. Além disso, foram apresentados alguns exemplos práticos da modelagem objeto relacional entre tabelas, colunas e relacionamentos no banco de dados e suas respectivas representações através de classes, atributos e anotações de relacionamentos no Spring Boot.

## Atividade Extra

Se quiser obter mais informações sobre a utilização dos relacionamentos no Spring Boot, leia o seguinte tutorial:

Link: https://www.baeldung.com/hibernate-one-to-many **(Acesso em 14/09/2022)**

## Referência Bibliográfica

PUGA, Sandra, FRANÇA, Edson., GOYA, Milton. **Banco de Dados Implementação em SQL, PL/SQL e Oracle** 11g. São Paulo: Pearson, 2014.

Ir para exercício

